LA LIBRERIA OPEN MUSIC OMCHROMA DOCUMENTAZIONE ONLINE

Luca Richelli
Sound and Music Processing Lab Conservatorio "C. Pollini" - Padova
lucarichelli@gmail.com

SOMMARIO

OMChroma [1] è una libreria per la sintesi digitale integrata dentro l'ambiente di aiuto alla composizione OpenMusic [2]. È una generalizzazione di Chroma, un sistema sviluppato da Marco Stroppa all'inizio degli anni Ottanta al Centro di Sonologia Computazionale dell'Università di Padova e in seguito riscritto in Lisp ed integrato in OpenMusic a partire dal 2000 da Carlos Agon e Marco Stroppa [3].

Quest'articolo illustra le caratteristiche salienti della documentazione online di OMChroma [4], scritta dall'autore per incarico dell'Institute de Recherche et de Coordination Acoustique/Musique (IRCAM) di Parigi.

1. INTRODUZIONE

Esistono altre librerie per l'integrazione di Csound in OpenMusic, tra cui *Om2Csound* realizzata da Karim Haddad, Mikhail Malt e Laurent Pottier nel 1999 [5] e *om4Csound* sviluppata da Mauro Lanza dal 2008 [6]. Tutte e tre le librerie si basano sul motore di sintesi Csound [7] con il fine di integrare le potenzialità di sintesi sonora di Csound all'interno di OpenMusic, ma differiscono nell'impostazione concettuale.

Om2Csound è finalizzata solamente alla compilazione dei file *score* da utilizzare direttamente con Csound autonomamente. *Om4Csound* offre invece un sistema che integra il software di sintesi (Csound) all'interno del software di controllo usufruendo dei vantaggi della programmazione grafica ad oggetti (object-oriented visual programming language).

OMChroma condivide alcune caratteristiche con alcuni sistemi dedicati alla sintesi sonora, progettati intorno agli anni Novanta, come ad esempio *Formes* [8], *Common Music/Common Lisp Music* [9] e *Patchwork/PWGL* [10], il cui scopo è la creazione di un ambiente integrato per l'elaborazione di strutture di controllo per la sintesi sonora.

2. LA MATRICE DI OMCHROMA

Sebbene allo stato attuale sia implementato l'interfacciamento solamente con Csound, OMChroma permette la rappresentazione dei parametri di controllo della sintesi indipendentemente dal motore di sintesi attraverso una matrice bidimensionale.

In Csound, come nei precedenti *Music N* software [11], il processo di sintesi è diviso in due parti: un'*orchestra* (file .orc) [12] contenente uno o più *instrument* che realizzano il digital signal processing, e una *score* (file .sco) [13] contenente i parametri necessari all'*orchestra* per la sintesi seguendo una lista temporale (*timeline*) di singoli eventi. La comunicazione tra l'*orchestra* e la *score* avviene tramite i *p-field* [14] gestibili a piacere dall'utente, ad eccezione dei primi tre riservati dal sistema (*instrument number*, *start-time e duration*).

In OMChroma la stesura dell'orchestra consiste nell'inizializzare, all'interno dell'ambiente grafico, un'orchestra di Csound tradizionale, ovvero un file di testo. L'inizializzazione creerà una *user class* specifica per ogni strumento dell'*orchestra*: per esempio l'orchestra "myorchestra.orc" contenente tre strumenti genererà le *user class* "myorchestra-1, myorchestra-2 e myorchestra-3". Poichè il primo *p-field* delle *score* indica il numero strumento e le *user class* di OMChroma indicano già uno strumento determinato, gli slot per l'inserimento dei dati (*p-field*) delle *user class* saranno pari al numero dei *p-field* dichiarati nell'orchestra meno 1 (ovviamente il primo *p-field*).

Per quanto riguarda la *score* le classi di OMChroma rappresentano i dati attraverso una matrice ruotata rispetto alla *score* di Csound. Nella *score* Csound ad ogni riga corrisponde un evento singolo con tutte le sue componenti mentre in OMChroma ogni riga contiene tutti i dati riguardanti il medesimo *p-field*.
Prendendo ad esempio la seguente *orchestra* di Csound che realizza un'oscillatore sinusoidale controllato in ampiezza (*p-4*) e frequenza (*p-5*)

```
instr 1
iamp = ampdb (p4)
ifreq = p5
ifn = 1
asig oscil iamp, ifreq, ifn
out asig
endin
```

e la *score*

p-1	p-2	p-3	p-4	p-5
instr. #	init time	dur	amp	freq
1	0	3	0	440
1	3	1	-6	660
1	5	2	-12	880

Tabella 1. Score Csound

La corrispondente *user class* in OMChroma rappresenta i dati nel modo sequente:

p-2	e-del	0	3	5
p-3	dur	3	1	2
p-4	amp	0	-6	-12
p-5	freq	440	660	880

Tabella 2. Matrix OMChroma

3. OMCHROMA USER MANUAL

3.1 System Configuration and Installation

"System Configuration and Installation" riporta le informazioni necessarie per l'installazione della libreria OMChroma e sugli altri componenti necessari al corretto funzionamento della libreria all'interno di OpenMusic oltre ai link utili.

3.2 Getting started

In OMChroma gli elementi disponibili per la creazione della programmazione visuale (*patch*), sono rappresentati da contenitori (*box*) differenti, suddivisi in *class* (classi) e *function* (funzioni).

In OMChroma, come in OpenMusic, le classi sono delle strutture di dati, mentre le funzioni si riferiscono alla funzioni sottostanti che eseguono una determinata operazione. Per chiarire: in OpenMusic *TEXTFILE*, un buffer di testo, è una classe, mentre *OM+*, l'operazione addizione, è una funzione. Il capitolo passa quindi in rassegna, con l'ausilio della classe ADD-1 (sintesi addittiva) e della funzione *synthesize*, le procedure di editing di base.

Ogni classe di sintesi è anche detta evento, ed ogni evento ha un determinato numero di elementi, gli input degli eventi sono detti *slot*. I primi quattro *slot* di ogni classe sono detti globali e visualizzati in blu. Gli altri slot, il cui numero è specifico per ogni classe, sono detti addizionali e visualizzati in rosso.

Una delle caratteristiche rimarchevoli dell'ambiente di programmazione di OMChroma è il polimorfismo: ogni tipologia differente di dati (numeri singoli, liste, *Break Point Function*, nomi di funzione, *lambda function*) viene processata secondo la sua caratteristica agevolando notevolmente la programmazione e il flusso dei dati.

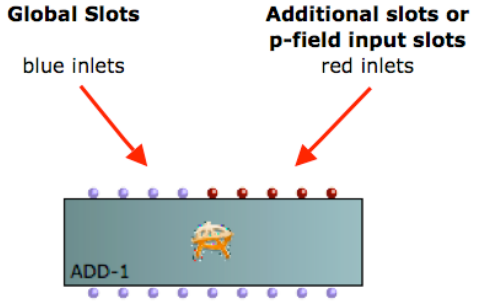

Figura 1. La classe ADD-1

3.3 Managing GEN function and sound files

In Csound è fondamentale la conoscenza e l'utilizzo delle *f-table*, vettori di valori a virgola mobile, generati dalle *GEN Routine* [15]. In OMChroma la gestione delle funzioni è uno degli aspetti cruciali per l'ottimizzazione del rendering audio e per sfruttare completamente le possibilità compositive di OpenMusic.

La libreria permette infatti un duplice impiego delle funzioni: locale e globale. Anche se è possibile ottenere lo stesso risultato usando sia le funzioni locali che quelle globali, il flusso di programma segue due percorsi differenti. Le funzioni globali vengono inizializzate solo una volta ed indirizzate tramite il loro numero univoco di assegnazione. Tutte le istanze di sintesi che faranno riferimento a questa funzione verranno indirizzate "fisicamente" alla stessa funzione.

Il funzionamento logico è quindi identico alla creazione di *f-table* all'interno della score di Csound. Le funzioni locali vengono ri-inizializzate per ogni singolo evento di sintesi, anche se la funzione rimane invariata. Di conseguenza è possibile un uso più flessibile a scapito di un maggior utilizzo di memoria ed una minore efficienza del rendering audio.

Il capitolo termina con la rassegna dei vari modi con cui la libreria può leggere i file audio.

3.4 Predefined Classes

OMChroma mette a disposizione 31 differenti classi predefinite con implementate le principali tecniche di sintesi. Lo scopo delle classi predefinite non è quello di comprendere tutte le tecniche di sintesi (impresa ovviamente impossibile), ma di mostrare all'utente una panoramica di strategie per la costruzione delle proprie classi (vedi in seguito 3.6 Creating a new Class) e di mettere a disposizione delle classi "pronte per l'uso" per incominciare a comprendere il funzionamento dell'ambiente di programmazione e le sue peculiarità.

Tecnica di sintesi	Elenco Classi
Additive Synthesis	ADD-1, ADD-2, ADD-3, ADD-A1
Buzz Synthesis	BUZZ-1, BUZZ-2, BUZFL-1
Frequency Modulation	FM-1, FM-2
Formant Wave Function (FOF)	FOF-1, FOF-2, FOF-3, FOF-4, FOF-A1, FOF-A4

Granular Formant Wave Function (FOG)	FOG-1
Karplus-Strong	PLUCK-1, PLUCK-2
Random Amplitude Modulation	RAN-1, RANFL-1
Sampler	SMPL-1, SMPL-2, SMPL-A1, SMPL-A2, SMPL-3, SMPL-4, SMPL-5, SMPL-6
Subtractive Synthesis	SUB-1
Wave Shaping	WSHP-1
Hybrid Models	SNARE-1

Tabella 3. Elenco delle classi predefinite

Sono allo sviluppo ulteriori classi, con il progetto di creare anche una sezione con quelle create dalla comunità degli utenti: OMChroma User Group (http://forumnet.ircam.fr/user-groups/).

3.5 User-fun

La *user function* è un potente mezzo per applicare una funzione lambda [16] ad ogni componente delle classi di OMChroma. In molti casi è possibile ottenere lo stesso risultato attraverso la programmazione delle funzioni di OpenMusic, ma la *user function* permette un maggior livello di astrazione ed una migliore efficienza di programmazione. La *user function* è una funzione che permette di effettuare una serie complessa di operazioni su tutta la matrice contenente le singole istanze di digital signal processing senza dover necessariamente procedere secondo la successione temporale degli eventi. Facendo riferimento al processo di sintesi di Csound, si ha la possibilità di intervenire con una funzione complessa per modificare, filtrare aggiungere, togliere, duplicare ogni singola istanza (*i-statement*) della *score* prima del rendering audio.

3.6 Creating a new Class

Questa è sicuramente la risorsa più grande della libreria. Anche se è possibile utilizzare OMChroma solamente con le classi predefinite, è fondamentale che l'utente abbia la possibilità di implementare le proprie classi (*user class*), che equivale alla scrittura delle proprie orchestre con Csound. Il procedimento non è particolarmente complesso: l'attenzione principale deve essere rivolta all'inizializzazione dei *p-field* utilizzati in funzione polimorfica (vedi 3.2 Getting started).

3.7 Multichannel processing

Tutte le classi predefinite sono monofoniche. Le *user class* possono ovviamente avere un'uscita multicanale utilizzando gli appositi *opcode* di Csound, ma c'è un modo più elegante e sofisticato per la gestione della disposizione spaziale dei file audio: l'utilizzo della libreria OMPrisma [17] ora integrata in OMChroma attraverso la funzione *Chroma-Prisma*.

OMPrisma è una libreria per il controllo dei processi di spazializzazione che permette di sviluppare complesse morfologie spaziali attraverso una descrizione simbolica dei parametri di sintesi. È dunque possibile tenere separati il processo di sintesi e quello di spazializzazione, permettendo la realizzazione di rendering alternativi in funzione del numero e delle possibili disposizioni degli altoparlanti previsti per la diffusione sonora.

4. SVILUPPI FUTURI

Il completamento della documentazione è attualmente allo stadio progettuale e prevede:

- Una breve introduzione alle tecniche di sintesi utilizzate nelle classi predefinite,
- L'estensione delle classi predefinite con l'inclusione di tecniche di sintesi granulari,
- L'aggiunta, alle classi predefinite, di una sezione di *classi avanzate* includendo anche quelle proposte dalla comunità degli utenti,
- Un capitolo dedicato al *Vertical Pitch Structure (VPS)* [18]: strutture polimorfiche che permettono di rappresentare i materiali spettrali come frequenze assolute, relative o in relazione simbolica ad una *frequenza-pivot*. Sono strutture che raccordano l'approccio compositivo armonico/melodico e quello numerico /spettrale,
- Un capitolo dedicato ai *Chroma-model*: strutture astratte di dati parametrici e formali per la sintesi, costituite sia da liste di segmenti temporali (time structures) che da sequenze di *Vertical Pitch Structure*,
- Un capitolo dedicato all'importazione delle analisi spettrali effettuate con Audiosculpt [19] o altri software che utilizzano il protocollo Sound Description Interchange Format (SDIF) [20] per la codifica dei dati.

5. CONCLUSIONI

La diffusione ed utilizzo di qualsiasi tipo di piattaforma software dipende, oltre che dalla validità intrinseca del progetto stesso e della sua realizzazione, anche dall'esistenza di una documentazione approfondita che permetta agli utenti di comprendere facilmente il funzionamento globale dell'ambiente e le funzioni di ogni singolo componente. Anche se la documentazione relativa ad una parte dei componenti di OMChroma non è ancora pronta, lo stato attuale del manuale permette all'utente di arrivare ad un buon livello di conoscenza della libreria e di essere in grado di sviluppare autonomamente le proprie *user class*: ovvero la risorsa principale di OMChroma.

La scelta di un'edizione online permetterà un aggiornamento continuo e costante del manuale, anche e soprattutto a seguito delle osservazioni della comunità degli utenti.

6. BIBLIOGRAFIA

[1] C Agon, J. Bresson, and M. Stroppa: “OMChroma: Compositional Control of Sound Synthesis” Computer Music Journal, 35:2 pp. 67–83, 2011.

[2] G. Assayag, C. Rueda, M. Laurson, C. Agon, and O. Delerue, "Computer Assisted Composition at Ircam : PatchWork & OpenMusicComputer" Computer Music Journal, 23:3, pp. 59-72 1999.

[3] C. Cadoz, A. Luciani, and J.-L. Florens: “Responsive input devices and sound synthesis by simulation of instrumental mechanisms: the Cordis system” Computer Music Journal, 8:3, pp. 60-73, 1984.

[4] http://support.ircam.fr/docs/om-libraries/omchroma/co/OMChroma.html visita 21 settembre 1014

[5] K. Haddad: "OpenMusic Om2Csound, Bibliothèque de modules de generation de scores pour Csound", version 1, IRCAM Centre Georges Pompidou, Paris 1999.

[6] M. Lanza, G. Verlingieri, and N. Biagioni: "La libreria OpenMusic om4Csound. Introduzione e progetto di documentazione" Atti del XVIII CIM Colloquio di Informatica Musicale Torino - Cuneo pp. 166-172, 2010

[7] B. Vercoe, "The Canonical Csound Reference Manual", Version 6.00.1, MIT http://csounds.com/manual/html/

[8] X. Rodet, and P. Cointe: "Formes: Composition and Scheduling of Processes". Computer Music Journal, 8:3, pp. 32-48, 1984

[9] H. Taube: "Common Music: A Music Composition Language in Common Lisp and CLOS". Computer Music Journal, 15:2, pp. 21-32, 1991

[10] M. Laurson, M. Kuuskankare, and V. Norillo: "An Overview of PWGL, a Visual Programming Environment for Music". Computer Music Journal, 33:1, pp. 19-31, 2009

[11] C. Roads: "The Computer Music Tutorial". Cambridge, Massachussetts: MIT Press, 1996

[12] R. Boulanger: "The Csound Book. Perspectives in Software Synthesis, Sound Design, Signal Processing, and programming".Cambridge, Massachussetts: MIT Press, pp. 7-8, 2000

[13] Ibid. pp. 8-9

[14] Ibid. pp. 14-15

[15] R. Bianchini, and A. Cipriani: “Il Suono Virtuale. Sintesi ed Elaborazione del Suono – Teoria e Pratica con Csound”. Seconda ed. Roma, ConTempo, pp. 9-15, 2001

[16] Bresson, C. Agon, and G. Assayag: "Visual Lisp/CLOS Programming in OpenMusic" Higher Order and Symbolic Computation vol. 22 pp. 81-111, 2009

[17] M. Schumacher, and J. Bresson: "Compositional Control of Periphonic Sound Spatialization" Proc. of the 2nd International Symposium on Ambisonics and Spherical Acoustics, May 6-7, Paris, 2010

[18] M. Stroppa: "Structure, Categorization, Generation and Selection of Vertical Pitch Structures (VPS). A Musical Application in Computer Assisted Composition" IRCAM Document, 1988 http://articles.ircam.fr/textes/Stroppa88a/index.pdf

[19] C. Diatkine: “AudioSculpt 3.0 User Manual”. 2011 http://support.ircam.fr/docs/Audio Sculpt/3.0/co/AudioSculptguideWeb. html

[20] D. Schwarz, and M. Wright: "Extensions and Applications of the SDIF Sound Description Interchange Format" International Computer Music Conference Berlin, pp. 481-484, 2000